JÜLICH
FORSCHUNGSZENTRUM

Deeper and active I/O hierarchies

D. Pleiter | CluStor Workshop | 30 July 2015

Outline

- Architectures and technologies
- Use cases for scientific computing
- Summary and conclusions

Architectures and Technologies

Non-volatile memories: opportunities

High bandwidth

- SATA 3.0: O(0.5) GByte/s
- PCIe GEN3: O(2-3) GByte/s

I/O versus memory style interface

- Continue to exploit established
  - I/O interfaces
  - Memory capacity management schemes
- Opportunity to use storage capacity as addressable memory

Small form-factor

- Flexible system integration
- Key for enabling active storage architectures

Non-volatile memories: limitations

Endurance

- O(5-10) full DWPD for high-end MLC NAND flash
- New technologies announced

High costs

- O(5) EUR/GByte for MLC NAND
  - Price drops expected
- Market primarily drives towards large capacity, not necessarily high bandwidth and high endurance
  - Development does not meet HPC requirements

→ **Deeper I/O hierarchies including small capacity, high-end NVM layers**

Selected active storage concepts

Active Disk [Acharya et al., 1998; Riedel et al., 1998, 1999]

- Ansatz: Take advantage of processing power on individual disk drives to run application-level code

Intelligent solid-state drives [Sangyeun Cho et al., 2013]

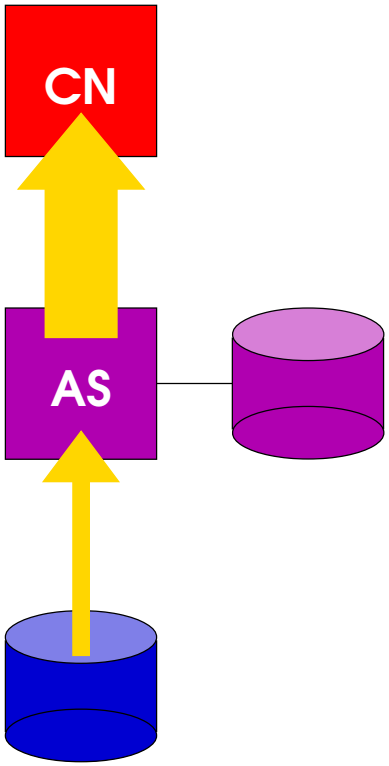
- Ansatz: Use embedded CPU which, e.g., implements FTL for data processing
- Potential advantage: internal bandwidth often higher

Active Storage in parallel file systems [Nieplocha et al., 2006, 2010]

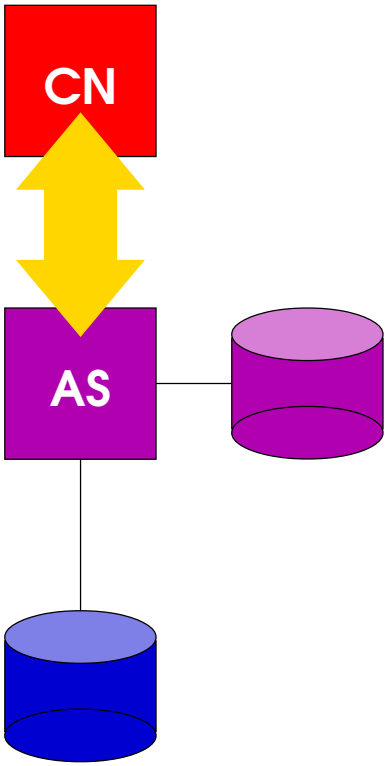
- Ansatz: Leverage processing power in server nodes of the parallel file system

And others ...

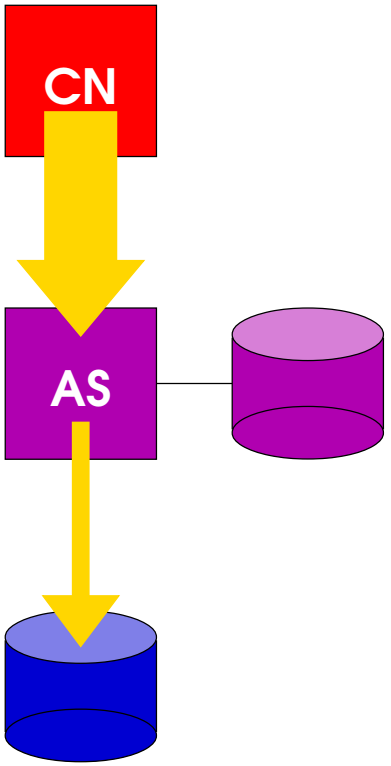
Use case classification: Data flow



Multi-pass analysis

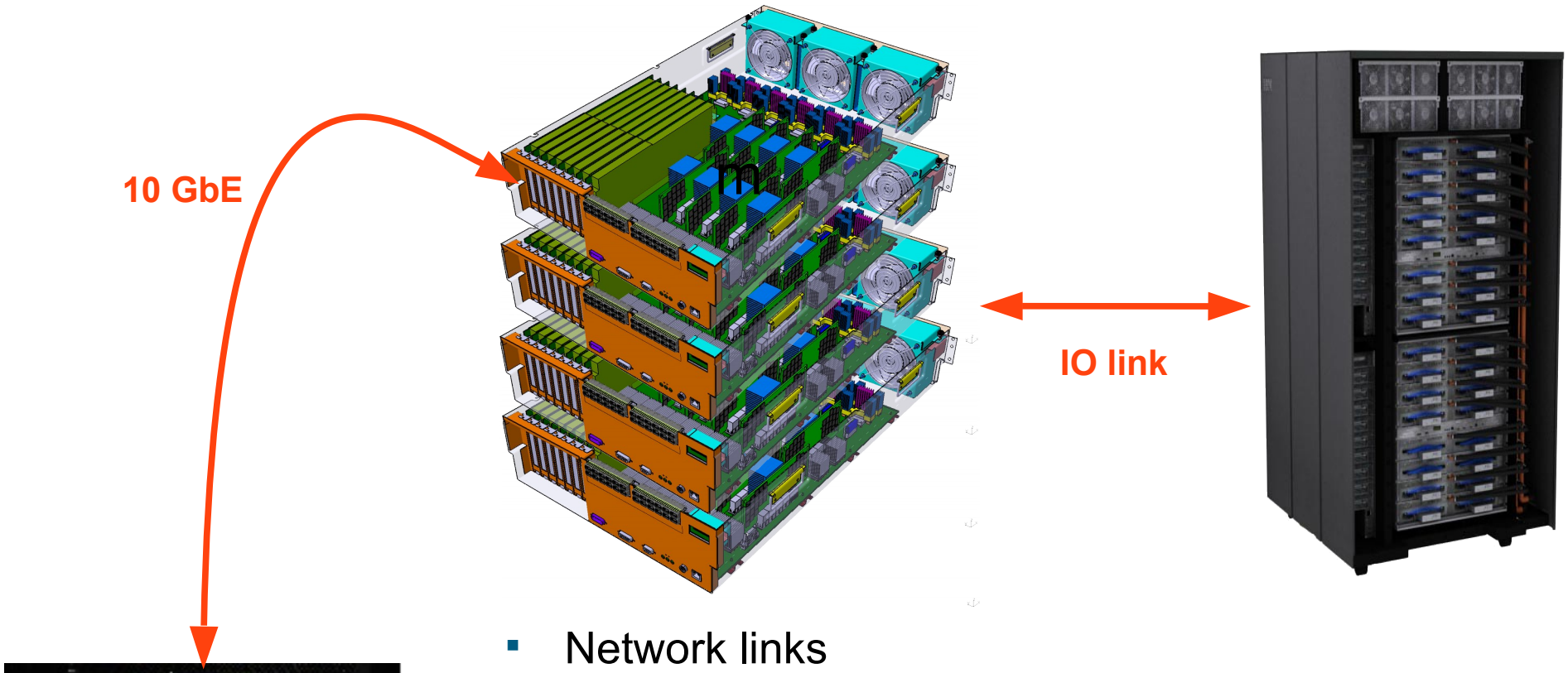


Out-of-core / check-pointing



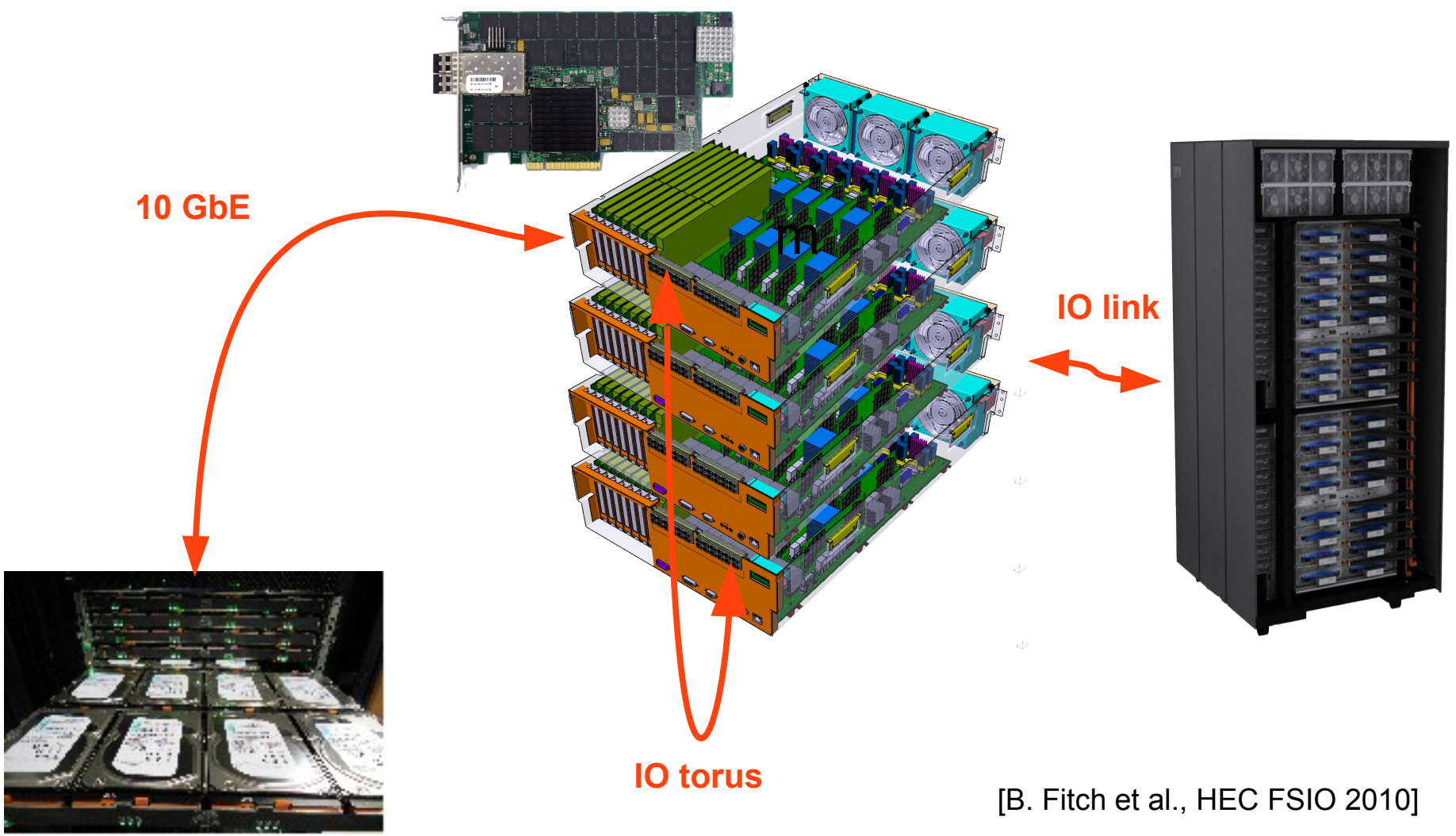
Post-processing / visualisation / check-pointing

Blue Gene/Q I/O architecture



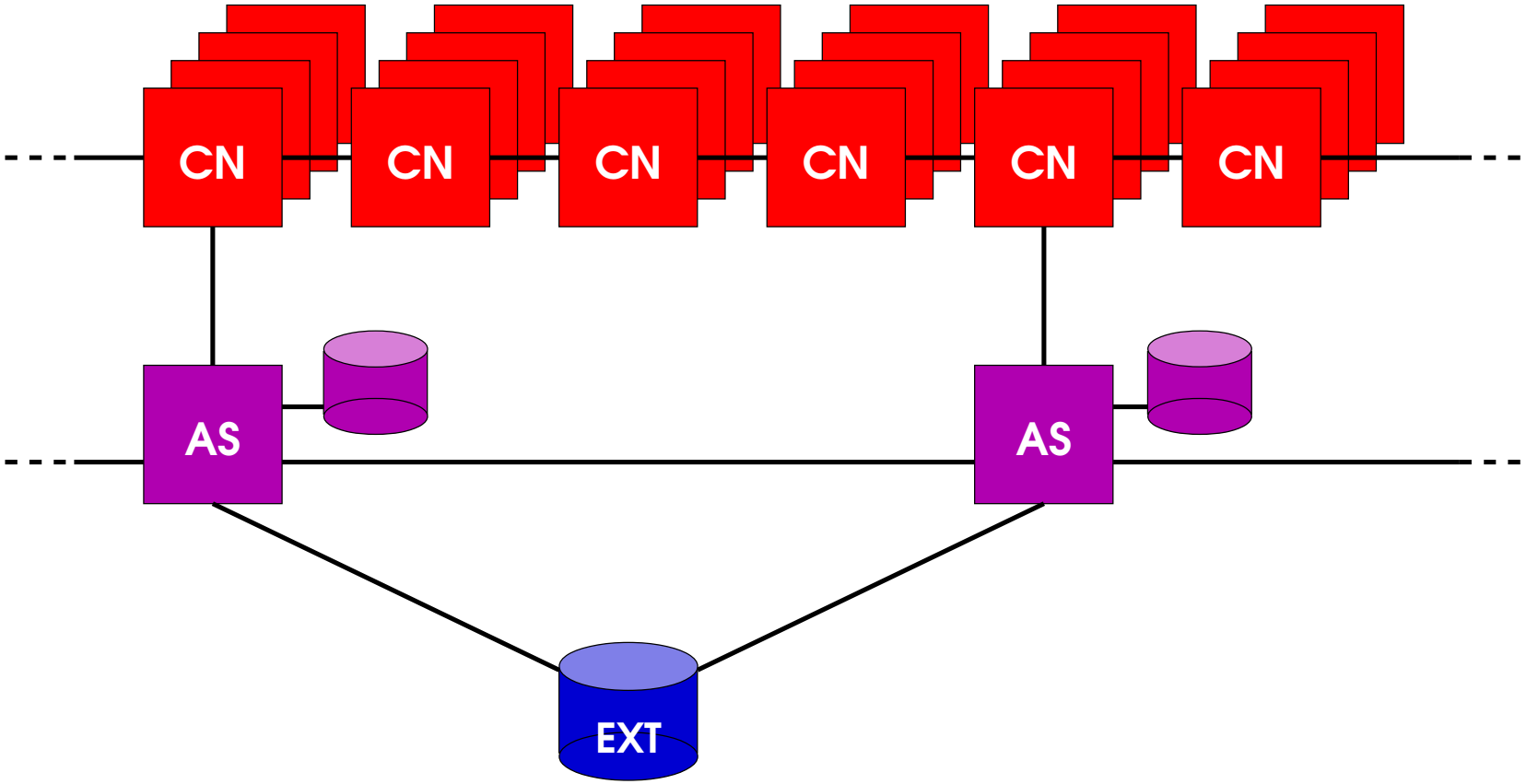
- Network links
  - 2 compute nodes connected to 1 I/O node
  - 2 10 GbE interfaces attached to 1 I/O node
- Forwarding of I/O functions
  - Slimline OS on compute nodes: CNK
  - Full Linux running on I/O nodes

Blue Gene Active Storage architecture



[B. Fitch et al., HEC FSIO 2010]

Blue Gene Active Storage architecture (cont.)



HS4 Card

Host interface

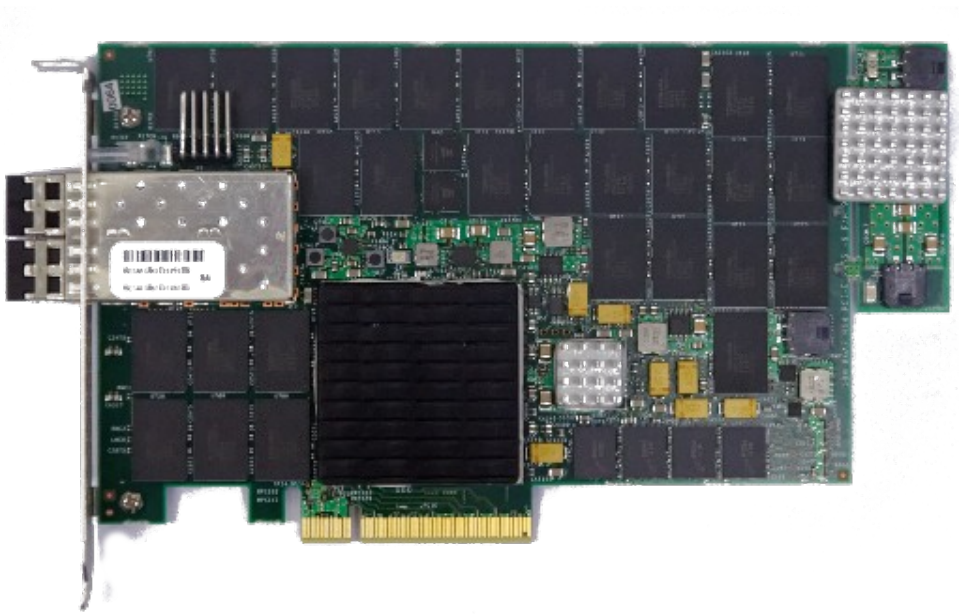
- x8 PCIe GEN2

Network interface

- 2x10 GE

Non-volatile memory

- 2 TByte SLC NAND flash
- MRAM



Digression: Burst buffer approach

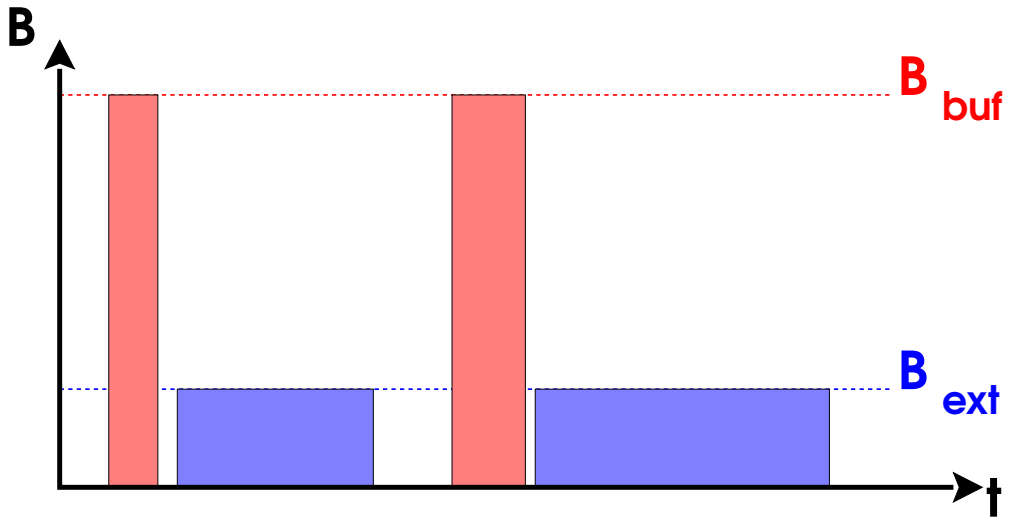
Observations

- Failure to scale I/O bandwidth
  - While demand may grow due to need for check-pointing
- Bursty I/O (write) patterns

Idea: introduce intermediate, high-bandwidth storage layer

[Bent and Grider, 2011]

Optimize bandwidth ratio by “impedance matching“



I/O behaviour

Long-term collection of GPFS performance counters

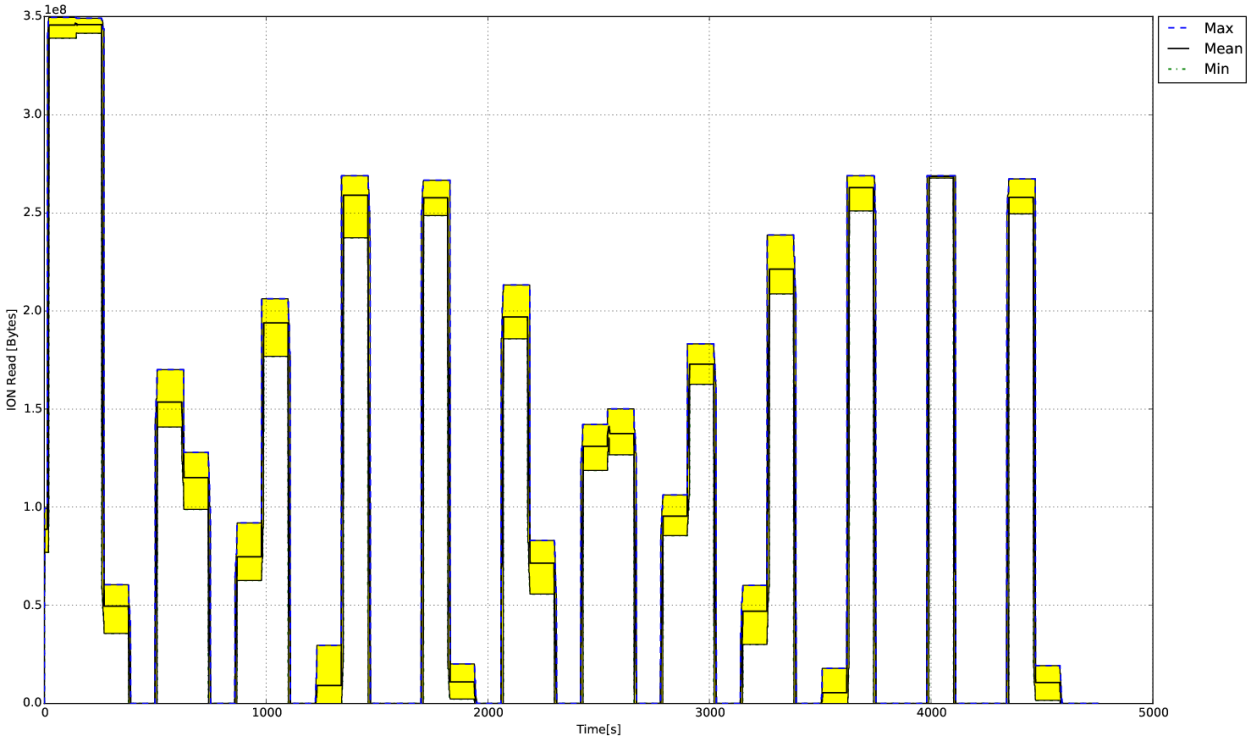
- Coarse-grained measurements at file system level

Observe significant fraction of jobs where

- Write patterns are not/not sufficiently bursty, or
- Read operations dominate

Example

- LQCD job on 1024 CN
- Read via 8 ION



[El Sayed, 2015]

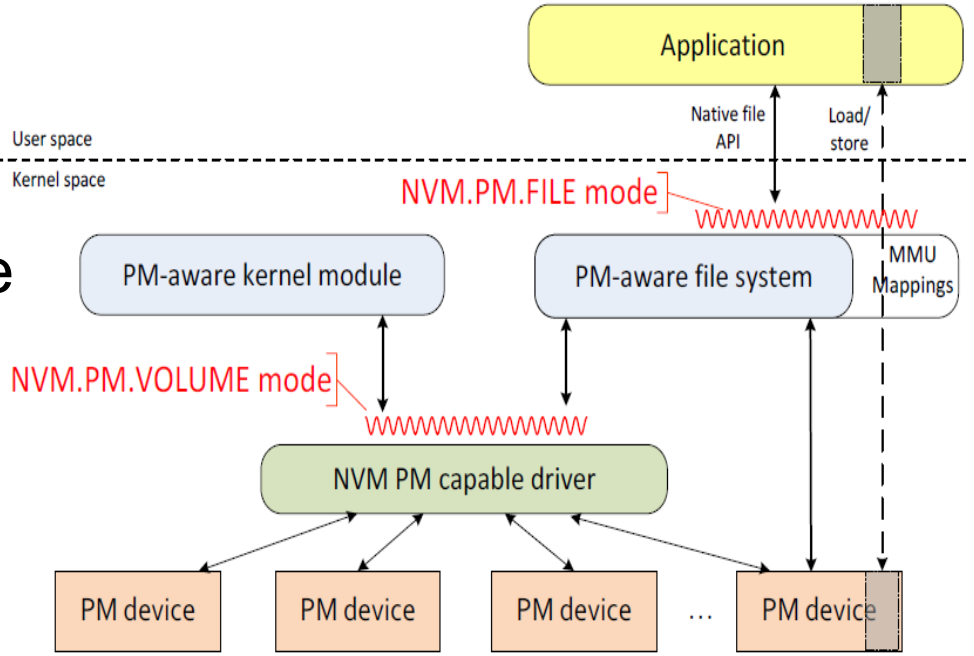
Digression: NVM programming models

File access

- Applications do not have to be aware of block storage hardware or addresses
- Established solutions for access control
- SNIA model: NVM.FILE

Addressable memory

- Memory address range
- associated with NVM volume
- Possibly MMU supported address translation
- SNIA model: NVM.PM.FILE



IBM's DSA interface

DSA = Direct Storage-class memory Access

- SCM = Byte addressable large capacity memory

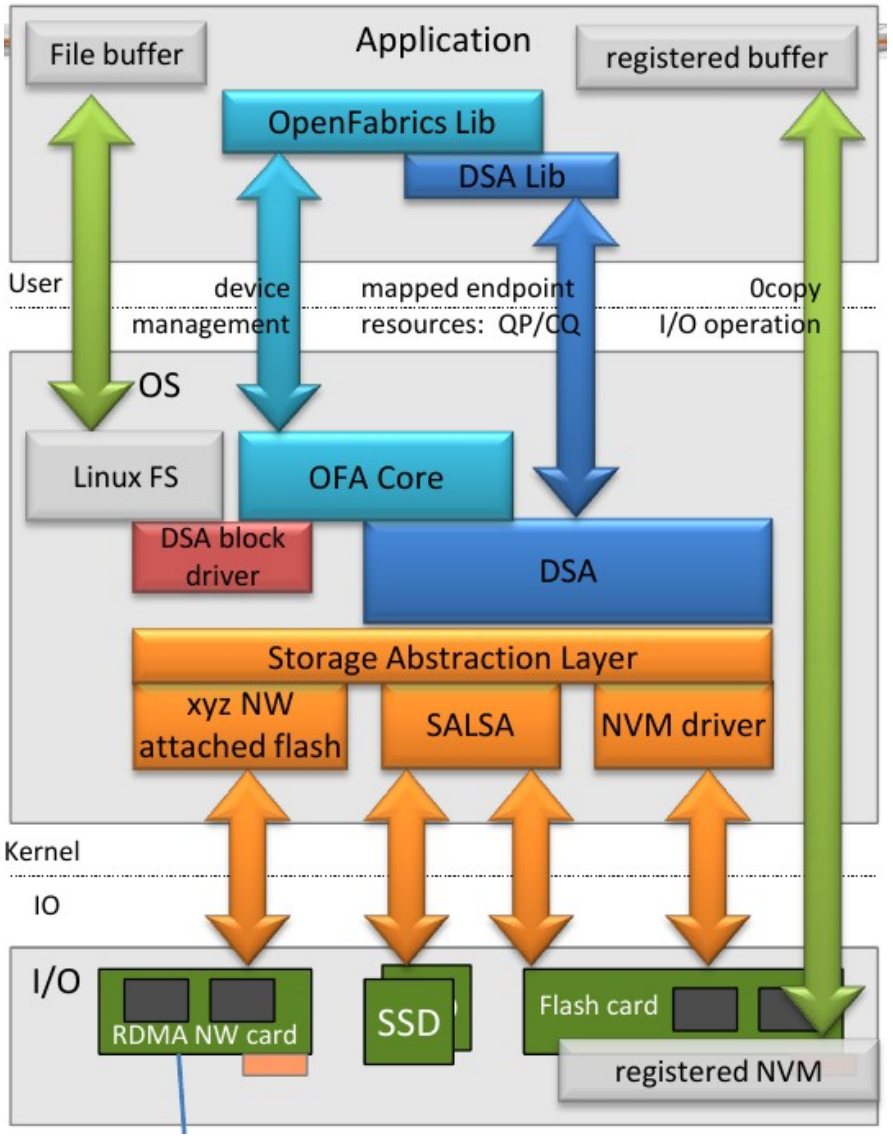
Idea: Use OpenFabrics stack for (local) NVM access

- HS4 card = verbs provider
- RDMA write/read to/from HS4

Benefits and features

- Byte addressable access to NVM
- Established verbs API
- Asynchronous data management
- OS by-pass, no block layer required
- Low CPU-load (function offload)
- Path towards full network integration

DSA interface (cont.)



Setup

- Open device and create protection domains
- Register memory
- Create queue pairs
- ...

Communication

- Post send or receive command
- Poll completion queue

[B. Metzler, 2015]

Digression: DSA vs. NVMe

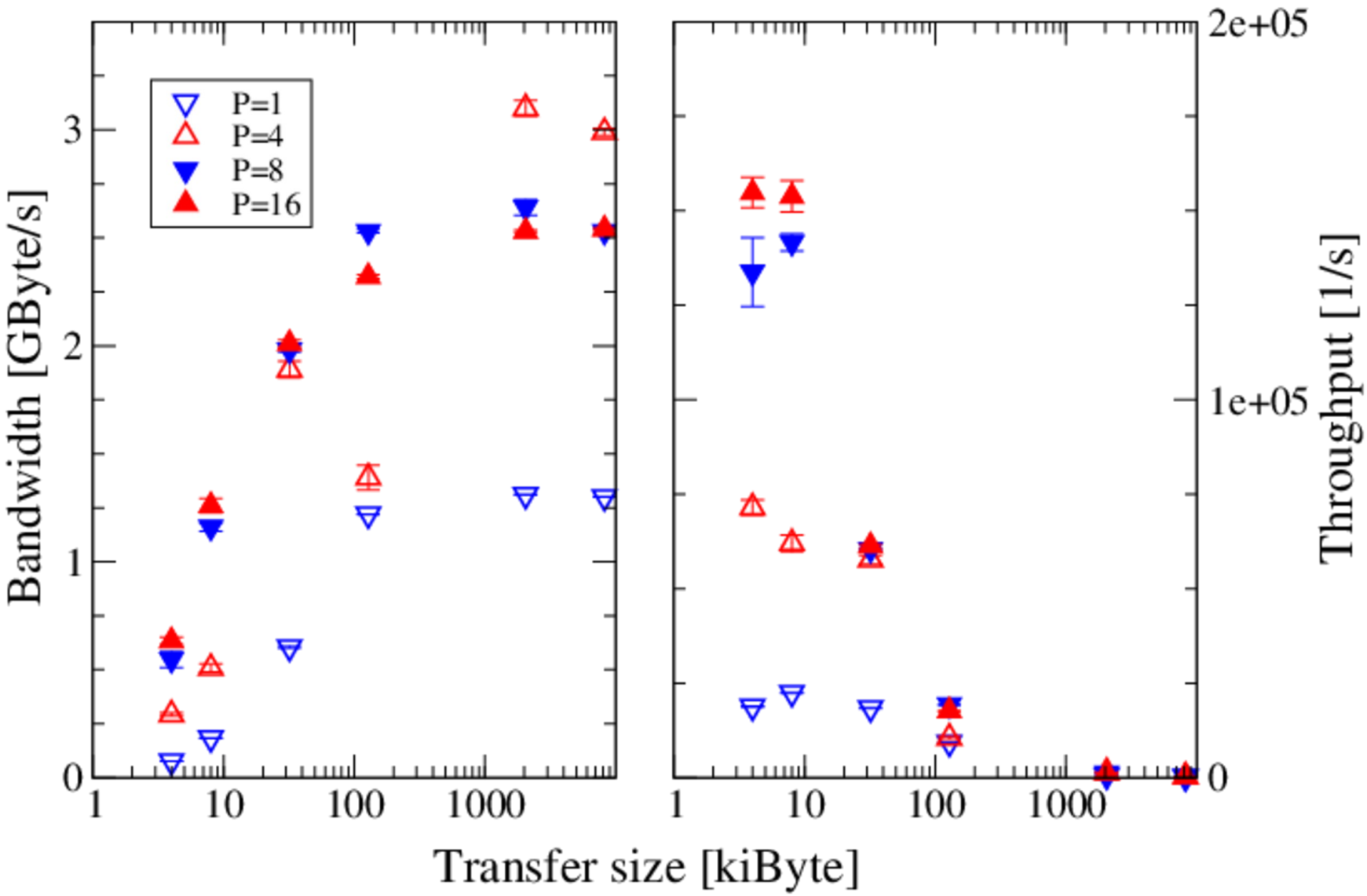
Similarities

- Data movement managed by controller
- Interface to controller via queue pairs

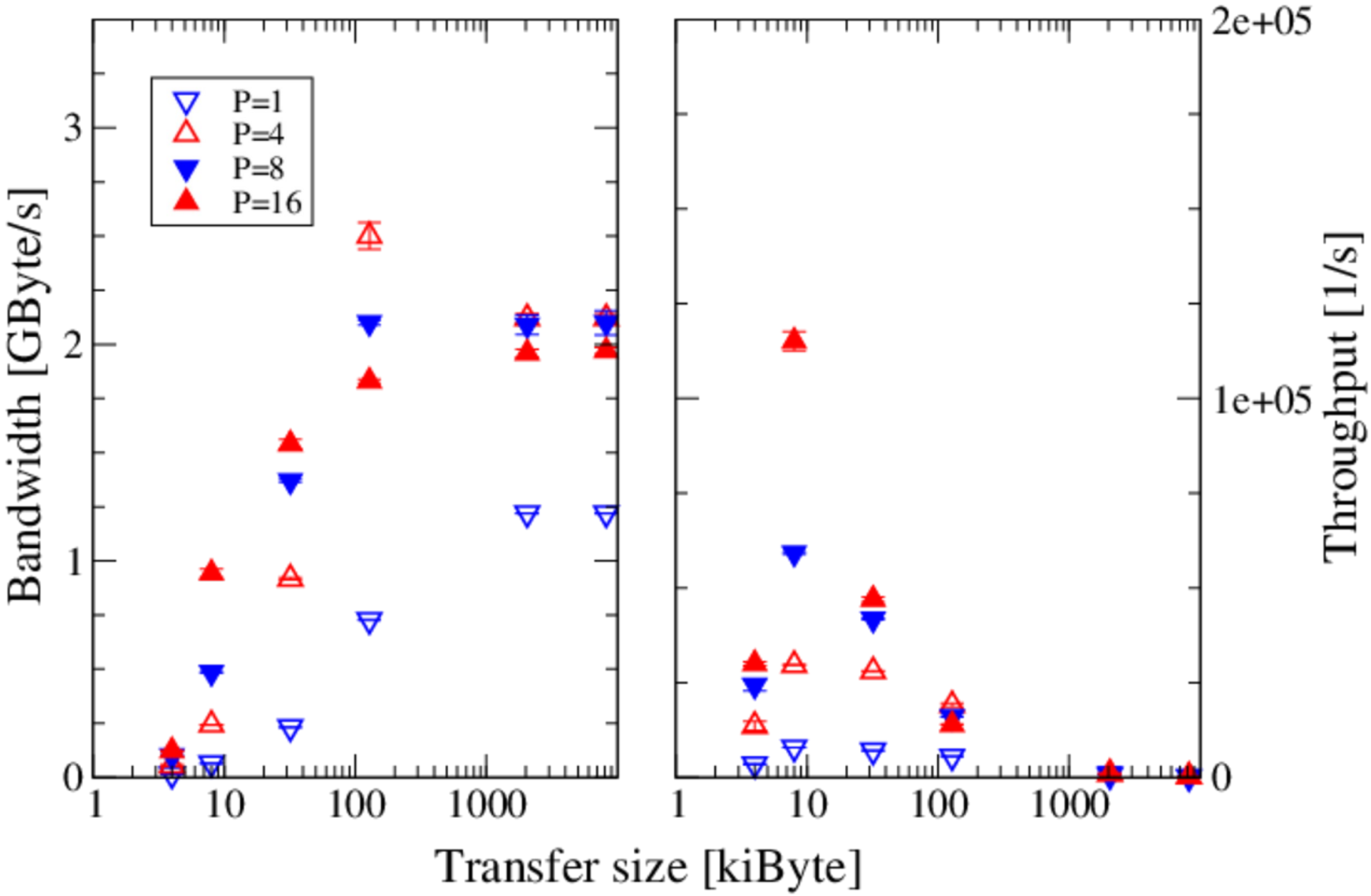
Differences

- Addressing
  - DSA: Continues address space
  - NVMe: Physical Region Page, Scatter Gather List
- OS integration
  - DSA: Above block layer, i.e. block layer can be by-passed
  - NVMe: Below block layer
- Network integration
  - DSA: Anticipated
  - NVMe: Unclear but proprietary solutions available

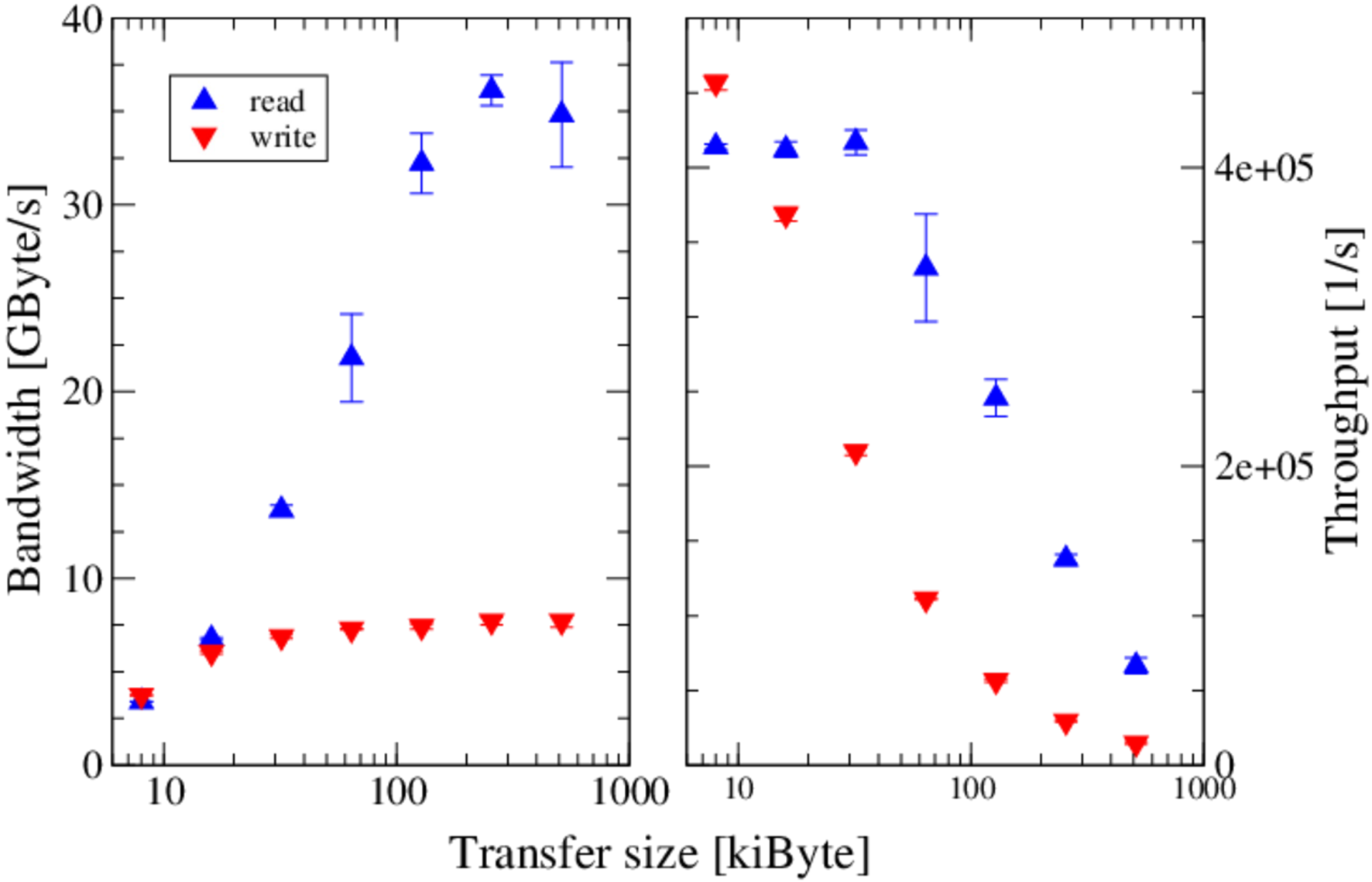
DSA micro-benchmark results: read



DSA micro-benchmark results: write

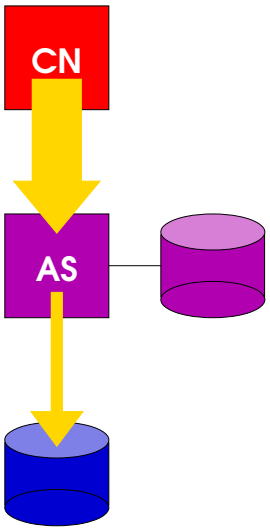


IOR running on CN (16 BGAS nodes)



Use Cases

Use case 1: Data post-processing



Assumed application characteristics

- Simulation which is generating significant amount of data
- Data volumes are too large to be written to disk
- Post-processing is data-intensive and non-scalable

Possible benefits

- Higher efficiency due to architecture optimized for data-intensive computing
- Interactive access easier to realise
  - Data can be kept available thanks to NVM

Spiking neuronal network simulator NEST

nest:: initiative

Application target

- Create models of the brains of mammals and humans
- Push limits of large-scale simulations of biologically realistic neuronal networks

Simulation challenge

- Size of the brain
  - Human cortex comprises about 10^{11} neurons
- High connectivity
  - On average there are about 10^{4} synapses per neuron

Data post-processing challenge

- O(10) GByte per biological s for spiking data only
- Membranes, synaptic weights, ... → 10,000x more data
- Interactive exploration of data

Jülich BGAS run-time system

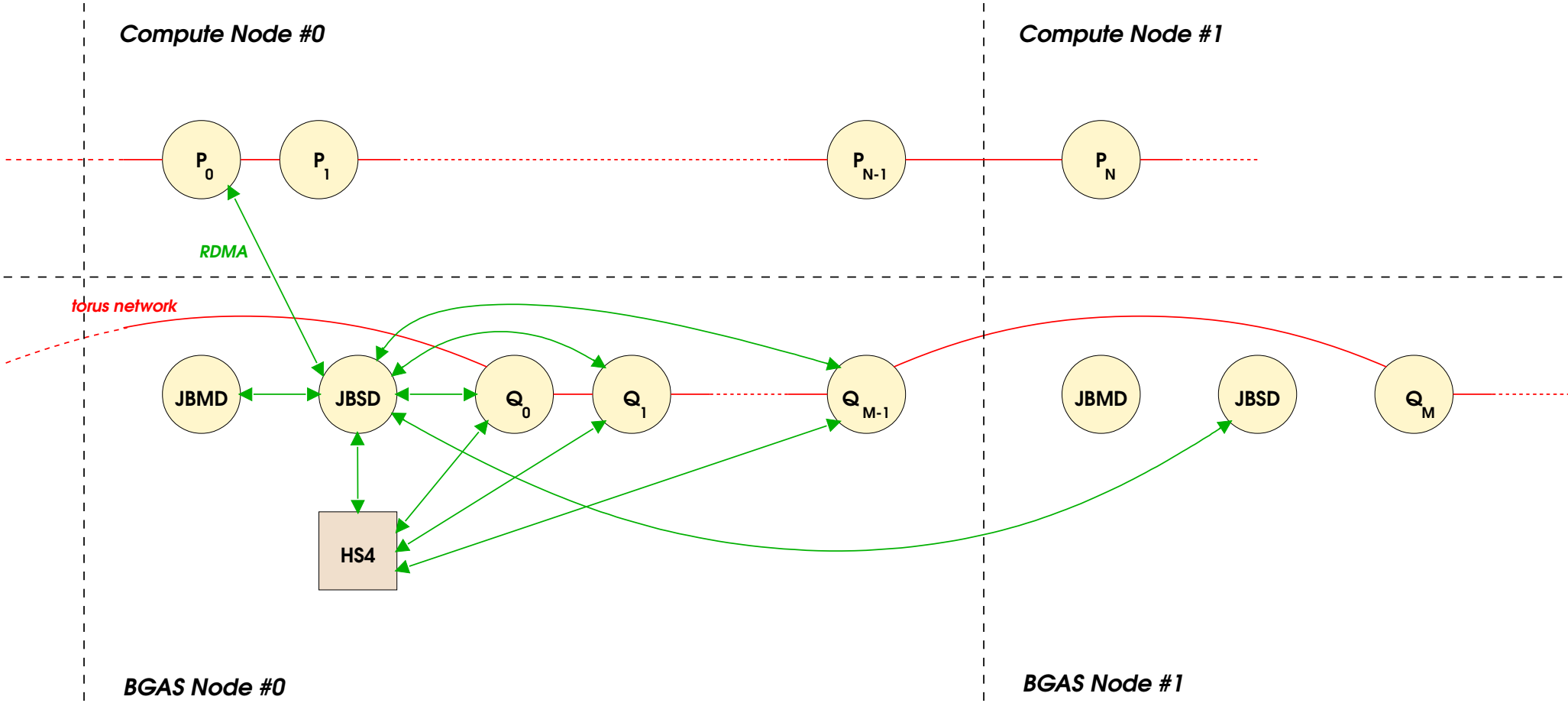
Required functionality

- Spawning of (user) processes within active storage
- Manage data transport
  - CN - ION via CNVerbs
  - ION - ION via RoQ
  - ION - HS4 via DSA

Design decisions

- Typically CN controlled environment
  - Allows to avoid co-scheduling challenge
- Aim for explicit locality management
  - Minimize "horizontal" traffic

Jülich BGAS run-time system (cont.)



Application call-outs

Challenges

- Heterogeneous architecture, different control domains
  - Different OS on CN and ION

Design choices

- Simple message communication to inform service daemon about request
- Service daemon starts launcher which spawns user process
  - Launcher remains connected to service daemon
- Simple notification mechanism
  - CN process has to poll for notifications

Use case 2: Visualisation

Large-scale data visualisation challenges

- Data volumes becoming large
  - Large = O(10) TByte, at least >4096³ voxels
  - Need capacity optimized memory technologies
- Data processing done in high-performance GPUs
  - High-performance = O(1) TFlop/s
  - Need for bandwidth optimized memory technologies

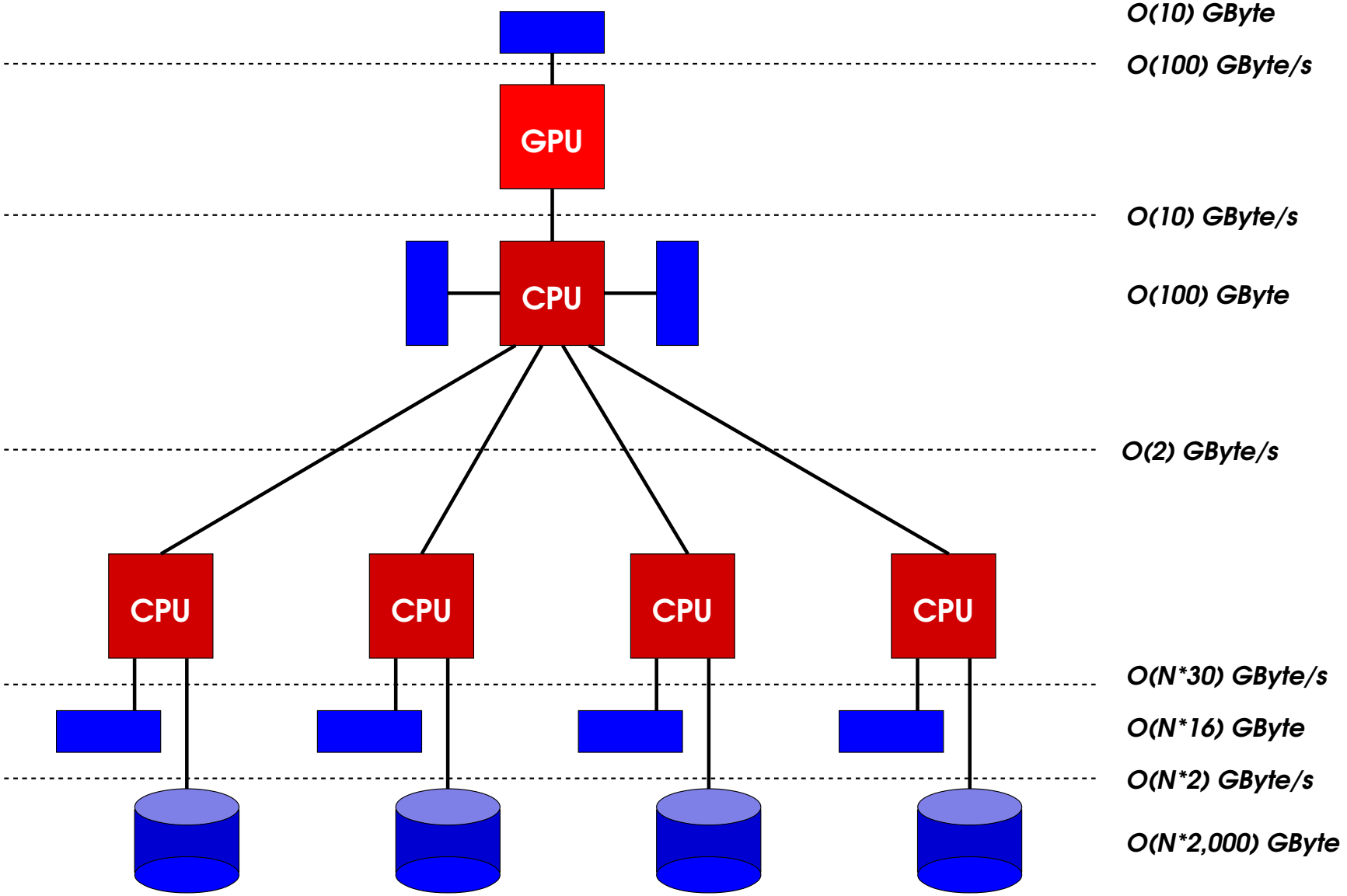
Idea: Software managed virtual memory hierarchy

- Proposed by M. Hardwiger et al. (2012)

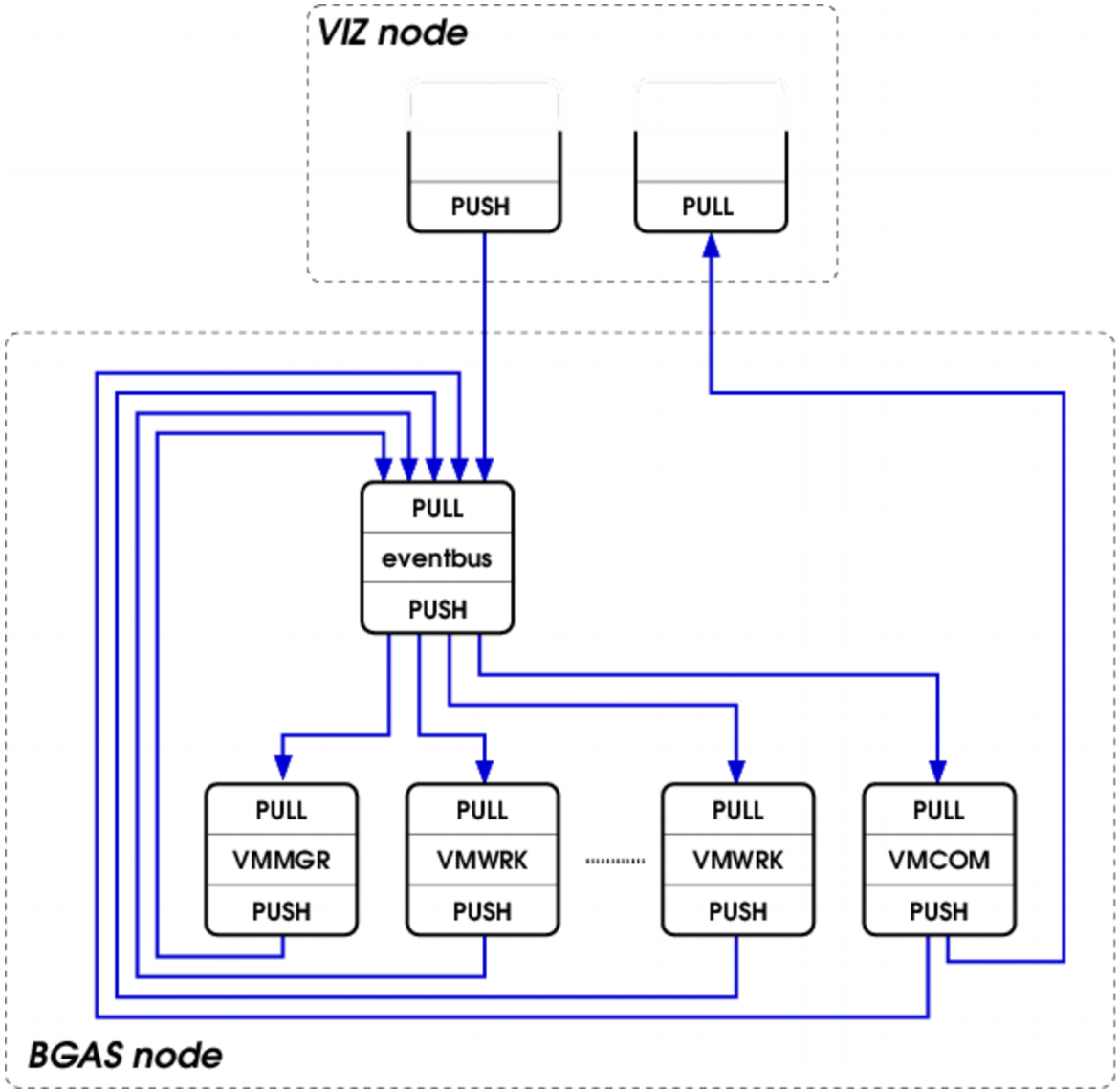
New aspects addressed together with RWTH Aachen

- Parallelisation of lowest layer
- Pre-processing of data at different levels
- Integration with HPC architecture

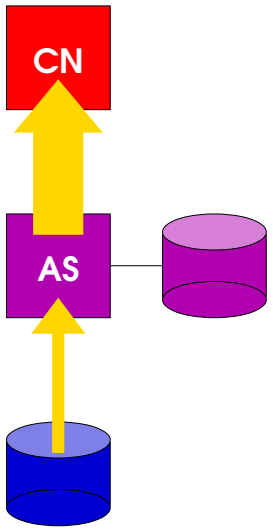
Virtual memory hierarchy



Visualisation on BGAS architecture



Use case 3: Data pre-fetching



Assumed application characteristics

- Large amount of external data to be processed
- Simple work-flow
  - Fetch data object
  - Process data
  - Write (small amount of) results
- Concurrent processing of data objects → many tasks
- Data fetching and processing can be overlapped

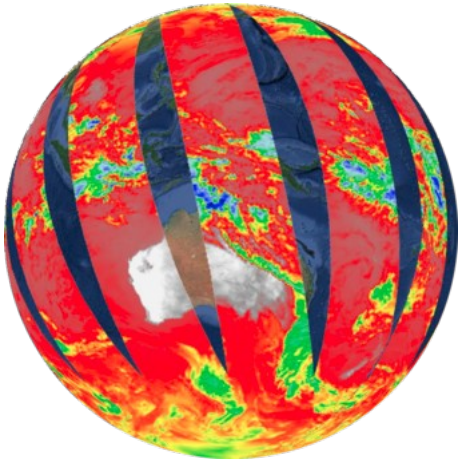
Possible benefits

- Avoid stalling compute nodes waiting for data
- Enable simple data pre-processing (reformatting)
- Reduce costs for multi-pass analysis
- Allow for less bandwidth to external storage

Satellite data processing

JURASSIC = Juelich Rapid Spectral Simulation Code

- Fast radiative transfer model for the infrared spectral region in the Earth's atmosphere
- Used for analysing ESA and NASA satellite
  - Satellites provide O(10...1000) TByte of measurement data during their lifetime



Computational characteristics

- Interpolation of data from pre-loaded look-up tables

I/O characteristics

- Data pre-processing (format conversion)
- Large number of small read accesses from HPC systems
- Data post-processing (format conversion)

Semi-persistent active cache

Idea: Allow storage to manage data object staging

- Cache = Storage holding temporary data object copies
  - Data object may be evicted if necessary
- Semi-persistent = Cache content may survive job boundaries
- Active = Support pre- and post-processing in cache

Work-flow

- Cache stages data objects depending on availability
- Compute nodes are informed about stage objects
- Optional mechanism for result file migration

Challenges

- Pre-fetch strategy
- Load balancing

Summary and Outlook

Summary and conclusions

Exciting technical and architectural road-maps enabled through non-volatile memories

- Different architectural concepts have been realised/are being realised
  - Burst buffers → **deeper I/O architecture**
  - Blue Gene Active Storage → **active I/O architecture**
- Major challenges ahead
  - System software-level integration
    - Global access interfaces
    - Management of extra storage
  - Programming models which hide heterogeneity and allow for portability

Different uses cases for scientific computing identified

- Show-cases for bringing large data volumes and HPC compute capabilities together

Outlook

Other use cases

- GVR = Global View Resilience
  - Project led by U Chicago/ANL
  - Aim for portable, application-controlled resilience
  - Error-handling interface supports error checking
    → Need for in-storage processing capabilities

Other technologies/projects

- DDN Infinite Memory Engine (IME)
  - Early incarnation of burst buffers
- SAGE project
  - H2020 funded FETHPC project
  - Starts in September 2015